kuando HTTP Manual

Version: 1.0.6

Content

1. Introduction

The kuando HTTP offers you a new application scenario to your kuando Busylight. Applications can now control your Busylight with http requests.

Kuando HTTP comes in two different flavours:

- One program running in the tray icon, as we know it from all the other integrations.
- One program running as Windows Service, invisible for the desktop user.

Please note: Do NOT use kuando HTTP with kuandoHUB at the same time. If you use kuandoHUB, the HTTP functionality is already built in!!

If you need support for kuando HTTP, please contact support@busylight.com.

For more information or tools to help with the implementation go to https://www.plenom.com.

2. HTTP API interface

The HTTP interface allow you to control the Busylight from your own software.

The HTTP Server access token is only required for remote access from other devices.

The HTTP server URL can only be set from the Registry. If the value cannot be processed, the http server will be disabled. The default value, http://localhost:8989/ will allow connections from the same machine. If you want to allow remote access, change the registry value to http://+:8989/.

For details about the HTTP server URL, see here: https://docs.microsoft.com/en-us/dotnet/api/system.net.httplistener

The http server access token is only used if a request is coming from a remote server.

Multiple Device Operation

If you connect more than a single Busylight device to your computer, you can choose to control all connected devices by the same commands from kuandoHUB, or you can control each single Busylight device from the HTTP interface.

The GET and the POST interface are capable to control a single Busylight. You need to add a path property to the GET parameters or the POST body JSON.

The path property needs to contain the values provides in the HIDDDevicePath string you can query with the busylightdevices command.

IMPORTANT!!!!

Please be aware that the strings containing the path cannot put into the GET-URL or the POST-JSON without conversion to make sure the content will reach the HTTP server completely.

For HHTP GET, you need to Url-Encode the path string.

For HTTP POST, you need to escape the special characters according to the JSON specification.

The details depend on the programming language you use.

For Powershell, you can use this snipplet for the GET command:

```
[System.Web.HttpUtility]::UrlEncode($Devices[0].HIDDevicePath)
```

For POST,l you can use

```
$path1= $Devices[0].HIDDevicePath | ConvertTo-Json
```

Please see the PowerShell example scripts for a complete example.

If you use path parameter with kuandoHUB, there will be a warning in the priorities page that not all devices are controlled by the priorities settings any more.

PowerShell Sample

HTTP GET example

```
# get connected Busylights

$devicesjson=Invoke-WebRequest -URI "http://localhost:8989?action=busylightdevices"

$devices = ($devicesjson.Content | ConvertFrom-Json)

if ($devices.Count -ge 2)
{


$path1="http://localhost:8989?action=light&red=100&green=100&blue=100&path="+[System.Web
.HttpUtility]::UrlEncode($Devices[0].HIDDevicePath)

    write-host "Testing Busylight "$path1

    Invoke-WebRequest -URI $($path1)


$path2="http://localhost:8989?action=light&red=0&green=100&blue=100&path="+[System.Web.H
ttpUtility]::UrlEncode($Devices[1].HIDDevicePath)

    write-host "Testing Busylight "$path2

    Invoke-WebRequest -URI $($path2)


}
else
{
    write-host "Please connect two Busylight Devices"
}
```

HTTP PUT example

```
# get connected Busylights

$devicescmd = '{"action":"busylightdevices"}'
$devicesjson=Invoke-WebRequest -Body $devicescmd -Method 'POST' -Uri
'http://localhost:8989'

$devices = $devicesjson.Content | ConvertFrom-Json

# Now: Testing...

if ($devices.Count -ge 2)
{
    write-host "Testing Busylight Devices"

    $path1= $Devices[0].HIDDevicePath | ConvertTo-Json

    $lightcmd =
'{"action":"Light","sender":"SDK","eventtype":"Light","eventname":"Color","parameter":"{
\"RedRgbValue\":0,\"GreenRgbValue\":100,\"BlueRgbValue\":0}", "path": ' + $path1  +' }'

    Invoke-WebRequest -Body $lightcmd -Method 'POST' -Uri 'http://localhost:8989'

    $path2= $Devices[1].HIDDevicePath | ConvertTo-Json

    $lightcmd2 =
'{"action":"Light","sender":"SDK","eventtype":"Light","eventname":"Color","parameter":"{
\"RedRgbValue\":0,\"GreenRgbValue\":0,\"BlueRgbValue\":100}", "path": ' + $path2  +' }'

    Invoke-WebRequest -Body $lightcmd2 -Method 'POST' -Uri 'http://localhost:8989'


}
else
{
    write-host "Please connect two Busylight Devices"
}
```

There is a second mechanism to address the Busylight devices by using the deviceindex parameter.

Please note that the list is zero-based, the fist device in the list has the index 0.

PowerShell Sample

HTTP GET example

```
$path1="http://localhost:8989?action=light&red=100&green=100&blue=100&deviceindex=0”
write-host "Testing Busylight "$path1
Invoke-WebRequest -URI $($path1)
$path2="http://localhost:8989?action=light&red=0&green=100&blue=100&deviceindex=1”
write-host "Testing Busylight "$path2
Invoke-WebRequest -URI $($path2)
```

HTTP PUT example

```
write-host "Testing Busylight Devices"
$path1= $Devices[0].HIDDevicePath | ConvertTo-Json
$lightcmd =
'{"action":"Light","sender":"SDK","eventtype":"Light","eventname":"Color","parameter":"{
\"RedRgbValue\":0,\"GreenRgbValue\":100,\"BlueRgbValue\":0}", "deviceindex": 0'
Invoke-WebRequest -Body $lightcmd -Method 'POST' -Uri 'http://localhost:8989'
$path2= $Devices[1].HIDDevicePath | ConvertTo-Json
$lightcmd2 =
'{"action":"Light","sender":"SDK","eventtype":"Light","eventname":"Color","parameter":"{
\"RedRgbValue\":0,\"GreenRgbValue\":0,\"BlueRgbValue\":100}", "deviceindex": 1 }'
Invoke-WebRequest -Body $lightcmd2 -Method 'POST' -Uri 'http://localhost:8989'
```

HTTP GET API interface

The HTTP GET interface is a very simple interface to control the Busylight. A http get command can be sent from an internet browser or a webhook as an example.

Below are the GET Requests that can be made: The examples show local requests which do not need a security token.

If a security token is needed, it is added as an additional parameter (http_token=tokendata)

If you want to use the default value, then you do not need to provide a parameter.

Light command:
`http://localhost:8989?action=light&red=100&green=100&blue=100`

Parameters:

Parameter Name	Values
Red	0..100 (Default 0)
Green	0..100 (Default 0)
Blue	0..100 (Default 0)

Alert Command:

```
http://localhost:8989?action=alert&red=100&sound=5&volume=25
```

Parameters:

Parameter Name	Values
Red	0..100 (Default 0)
Green	0..100 (Default 0)
Blue	0..100 (Default 0)
Sound	0..8 (See list)
Volume	0..100 (See list)

Sound list:

Sound	Sound number
(No Sound)	0
Fairy Tale	1
Funky	2
Kuando Train	3 (Default)
Open Office	4
Quiet	5
Telephone Nordic	6
Telephone original	7
Telephone Pick Me Up	8

Volume can be set to these values:

Volume	Volume value
100%	100
75%	75 (Default)
50%	50
25%	25
Mute	0

Blink command
`http://localhost:8989?action=blink&blue=100`

Parameters:

Parameter Name	Values
Red	0..100 (Default 0)
Green	0..100 (Default 0)
Blue	0..100 (Default 0)
Ontime (0.1 seconds steps light on)	Default: 5 (0.5 Seconds)
Offtime (0.1 seconds steps light off)	Default: 5 (0.5 Seconds)

Jingle command
`http://localhost:8989?action=jingle&red=100&sound=3&volume=100`

Parameters:

Parameter Name	Values
Red	0..100 (Default 0)
Green	0..100 (Default 0)
Blue	0..100 (Default 0)
Sound	0..8 (See list)
Volume	0..100 (See list)

For Sound and Volume values, see the Alert command.

Pulse command
`http://localhost:8989?action=pulse&blue=100&red=100`

Parameters:

Parameter Name	Values
Red	0..100 (Default 0)
Green	0..100 (Default 0)
Blue	0..100 (Default 0)

ColorWithFlash command
`http://localhost:8989?action=colorwithflash&green=100&red=100&flashblue=100`

Parameters:

Parameter Name	Values
Red	0..100 (Default 0)

Green	0..100 (Default 0)
Blue	0..100 (Default 0)
flashred	0..100 (Default 0)
flashgreen	0..100 (Default 0)
flashblue	0..100 (Default 0)

Off command

```
http://localhost:8989?action=off
```

The Off-Command has no parameters associated with it.

currentpresence command

```
http://localhost:8989?action=currentpresence
```

This command will return the current presence as json string. It contains the running priority information and the priority present in my own priority, regardless if it is running or not.

PowerShell Sample

```
Invoke-WebRequest -URI "http://localhost:8989?action=currentpresence"
```

JavaScript Sample

```
const Http = new XMLHttpRequest();
    const url='http://localhost:8989?action=currentpresence';
    Http.open("GET", url);
    Http.send();

    Http.onreadystatechange = (e) => {
      console.log(Http.responseText)
      }
```

busylightdevices command

```
http://localhost:8989?action=busylightdevices
```

This command will return the list of all connected Busylight devices, including details like USB VID/PID, Model name, UniqueID (not supported with all models) and the HIDDevicePath.

The HIDDevicePath is necessary to use address a single Busylight device.

PowerShell Sample

```
Invoke-WebRequest -URI "http://localhost:8989?action=busylightdevices"
```

JavaScript Sample

```
const Http = new XMLHttpRequest();
    const url='http://localhost:8989?action= busylightdevices';
    Http.open("GET", url);
    Http.send();

    Http.onreadystatechange = (e) => {
      console.log(Http.responseText)
      }
```

HTTP POST API interface

The HTTP POST interface can be used by any software which is able to do a HTTP POST request, e.g. PowerShell, JavaScript inside a Browser or a desktop app.

The HTTP POST interface enables all SDK functionalities including registering Priority configuration and Data Sources.

The HTTP POST interface accepts json data.

Common Parameters

The json structure has four common parameters, which control the kuandoHUB operation.

Parameter Name	Values
action	Command for kuandoHUB
sender	Datasource for kuandoHUB priority operation
eventtype	Event Type for kuandoHUB priority operation
eventname	The name of the specific event for kuandoHUB priority operatrion
parameter	This property contains an embedded json structure with the parameters for the specific action. The structure is different and specific for each action. (See examples below.)

Light command:

```
{
            "action":"Light",
            "sender":"SDK",
            "eventtype":"Light",
            "eventname":"Color",
            "parameter":"{
                        \"RedRgbValue\":0,
                        \"GreenRgbValue\":100,
                        \"BlueRgbValue\":0
                        }"
}
```

Parameters:

Parameter Name	Values
RedRgbValue	0..100 (Default 0)
GreenRgbValue	0..100 (Default 0)
BlueRgbValue	0..100 (Default 0)

Alert Command:

```
{
            "action":"Alert",
            "sender":"SDK",
            "eventtype":"Alert",
            "eventname":"Alert",
            "parameter":"{
                        \"color\":{
                                    \"RedRgbValue\":255,
                                    \"GreenRgbValue\":0,
                                    \"BlueRgbValue\":0
                                    },
                        \"clip\":5,
                        \"volume\":50
                        }"
}
```

Parameters:

Parameter Name	Values
RedRgbValue	0..100 (Default 0)
GreenRgbValue	0..100 (Default 0)
BlueRgbValue	0..100 (Default 0)
Clip	0..8 (See list)
Volume	0..100 (See list)

Sound list:

Clip	Sound number
(No Sound)	0
Fairy Tale	1
Funky	2
Kuando Train	3 (Default)
Open Office	4
Quiet	5
Telephone Nordic	6
Telephone original	7
Telephone Pick Me Up	8

Volume can be set to these values:

Volume	Volume value
100%	100
75%	75 (Default)
50%	50
25%	25
Mute	0

Blink command

```
{
```

```
            "action":"Blink",
            "sender":"SDK",
            "eventtype":"Blink",
            "eventname":"Blink",
            "parameter":"{
                        \"color\":{
                                    \"RedRgbValue\":255,
                                    \"GreenRgbValue\":0,
                                    \"BlueRgbValue\":0
                                    },
                        \"ontime\":5,
                        \"offtime\":5
                        }"
}
```

Parameters:

Parameter Name	Values
RedRgbValue	0..100 (Default 0)
GreenRgbValue	0..100 (Default 0)
BlueRgbValue	0..100 (Default 0)
Ontime (0.1 seconds steps light on)	Default: 5 (0.5 Seconds)
Offtime (0.1 seconds steps light off)	Default: 5 (0.5 Seconds)

Jingle command

```
{
            "action":"Jingle",
            "sender":"SDK",
            "eventtype":"Jingle",
            "eventname":"Jingle",
            "parameter":"{
                        \"color\":{
                                    \"RedRgbValue\":255,
                                    \"GreenRgbValue\":255,
                                    \"BlueRgbValue\":0
                                    },
                        \"clip\":9,
                        \"volume\":50
                        }"
}
```

Parameters:

Parameter Name	Values
RedRgbValue	0..100 (Default 0)
GreenRgbValue	0..100 (Default 0)
BlueRgbValue	0..100 (Default 0)
Clip	0..8 (See list)
Volume	0..100 (See list)

For Sound and Volume values, see Alert command.

Pulse command

```
{
            "action":"Pulse",
            "sender":"SDK",
```

```
            "eventtype":"Pulse",
            "eventname":"Pulse",
            "parameter":"{
                        \"color\":{
                                    \"RedRgbValue\":255,
                                    \"GreenRgbValue\":255,
                                    \"BlueRgbValue\":0
                                    },
                        \"pulsesequence\":{
                                    \"Color\":{
                                                \"RedRgbValue\":255,
                                                \"GreenRgbValue\":255,
                                                \"BlueRgbValue\":0
                                                },
                                    \"Step1\":3,
                                    \"Step2\":21,
                                    \"Step3\":36,
                                    \"Step4\":50,
                                    \"Step5\":36,
                                    \"Step6\":21,
                                    \"Step7\":10
                                    }
                        }"
}
```

Parameters:

Parameter Name	Values
RedRgbValue	0..100 (Default 0)
GreenRgbValue	0..100 (Default 0)
BlueRgbValue	0..100 (Default 0)
Step1 .. Step7	Intensity of the step (0..100)

ColorWithFlash command

```
{
            "action":"ColorWithFlash",
            "sender":"SDK",
            "eventtype":"Light",
            "eventname":"ColorWithFlash",
            "parameter":"{
                        \"color\":{
                                    \"RedRgbValue\":0,
                                    \"GreenRgbValue\":255,
                                    \"BlueRgbValue\":0
                                    },
                        \"flash\":{
                                    \"RedRgbValue\":0,
                                    \"GreenRgbValue\":0,
                                    \"BlueRgbValue\":255
                                    }
                        }"
}
```

Parameters:

The color parameter determines the solid color, the color of the short flashes. Both colors are defined as designated in the parameter table:

Parameter Name	Values
RedRgbValue	0..100 (Default 0)

GreenRgbValue	0..100 (Default 0)
BlueRgbValue	0..100 (Default 0)

Off command

```
{
            "action":"Off",
            "sender":"SDK",
            "eventtype":"Light",
            "eventname":"Off",
            "parameter":""
}
```

The Off-Command has no specific parameters.

RegisterDataSource command

This command allows you to register a custom data source in kuandoHUB for priority operation. It is recommended to send this command on every connect to kuandoHUB to make sure that the data source exists and has the most recent version. See kuandoHUB operation chapter for details.

```
{
            "action":"RegisterDataSource",
            "sender":null,
            "eventtype":null,
            "eventname":null,
            "parameter":"{
                        \"DataSourceName\":\"SDK\",
                        \"eventnames\":{
                                    \"Light\":{
                                                \"EventNames\":[
                                                            \"Light\",
                                                            \"Green\",
                                                            \"Red\",
                                                            \"Yellow\",
                                                            \"Off\"],
                                                \"IsAlertPriority\":false
                                                },
                                    \"Alert\":{
                                                \"EventNames\":[
                                                            \"Alert\",
                                                            \"Other Notification\"],
                                                \"IsAlertPriority\":true
                                                }
                                    }
                        }"
}
```

CreateInitialPriority command

The CreateInitialPriority command allows to create Priority entries in the kuandoHUB priority operation.

```
{
            "action":"CreateInitialPriority",
            "sender":null,
            "eventtype":null,
            "eventname":null,
            "parameter":"{
                        \"enabled\":true,
                        \"sender\":\"SDK\",
                        \"eventtype\":null,
                        \"eventnames\":[],
```

```
                        \"IsAlertPriority\":false
                        }"
}
```

This command creates a line in kuandoHUB priorities with the data Source name “SDK” containing all eventtypes and event names.

The command will have no effect if a line containing the data source name is already in the kuandoHUB priorities list.

currentpresence command

```
{
            "action":"currentpresence",
            "sender":"SDK",
            "eventtype":"",
            "eventname":"",
            "parameter":""
}
```

The currentpresence-Command has no specific parameters.

This command will return the current presence as json string. It contains the running priority information and the priority present in my own priority, regardless if it is running or not.

Powershell sample

This command switches the Busylight to green light. Please make sure you have registered the SDK DataSource and have an enabled priority line entry in kuandoHUB.

```
$lightcmd = '{"action":"currentspresence","sender":"SDK"}'

Invoke-WebRequest -Body $lightcmd -Method 'POST' -Uri 'http://localhost:8989'
```

JavaScript sample

This commands switches the Busylight to green light. Please make sure you have registered the SDK DataSource and have an enabled priority line entry in kuandoHUB.

```
    const Http = new XMLHttpRequest();
    const url='http://localhost:8989';
    Http.open("POST", url);
    Http.send('{"action":"currentspresence","sender":"SDK"}');
    Http.onreadystatechange = (e) => {
      console.log(Http.responseText)
      }
```

Busylightdevices command

```
{
            "action":"busylightdevices"
}
```

This command returns the list of all connected Busylight devices, including details like USB VID/PID, Model name, UniqueID (not supported with all models) and the HIDDevicePath.

The HIDDevicePath is necessary to use address a single Busylight device.

Powershell sample

This command switches the Busylight to green light. Please make sure you have registered the SDK DataSource and have an enabled priority line entry in kuandoHUB.

```
$lightcmd = '{"action":"busylightdevices","sender":"SDK"}'

Invoke-WebRequest -Body $lightcmd -Method 'POST' -Uri 'http://localhost:8989'
```

JavaScript sample

This commands switches the Busylight to green light. Please make sure you have registered the SDK DataSource and have an enabled priority line entry in kuandoHUB.

```
const Http = new XMLHttpRequest();
const url='http://localhost:8989';
Http.open("POST", url);
Http.send('{"action":"busylightdevices","sender":"SDK"}');
Http.onreadystatechange = (e) => {
  console.log(Http.responseText)
  }
```

3. HTTP Responses

These response codes are defined in the Busylight HTTP implementations:

200 (OK)	Process request successful
404 (no found)	Busylight device is not connected
401 (unauthorized)	http token invalid

Appendix A – Registry settings

These registry keys are stored in the path `HKCU\Software\Busylight` and can be changed or added via registry key or centrally via group policies. Changes may be also applied after installation of the Busylight software.

The following table contains all registry keys, which are used by Busylight, along with the default values if the key does not exist:

Registry Key	Data type	Description	Default Value (Decimal)
http_uri	SZ	Listening URL	http://localhost:8989/
http_token	SZ	Access token for remote access	

The following table contains special registry keys:

Registry Key	Data type	Description	Default Value (Decimal)
SDKLogLevel	DWORD	Log level for USB communication 0: no messages 1: verbose logging	0

Appendix B – Group Policy settings

It is possible to predefine Corporate standards on any setting via group policy. The kuandoHUB app checks the `Software\Policies\Busylight` sections for existence of any of the registry keys listed in the previous chapter.

If a setting is found in the Policies sections, it will override any user setting and disable the corresponding user menu items. Users can still see the predefined settings, but not change them.

Upon application start, the registry sections will be checked in this order:

1. `HKEY_LOCAL_MACHINE\Software\Policies\Busylight`
2. `HKEY_CURRENT_USER\Software\Policies\Busylight`
3. `HKEY_CURRENT_USER\Software\Busylight`
4. `HKEY_LOCAL_MACHINE\Software\Busylight`

The first value found in this order will be applied. If nothing is found in the registry, kuandoHUB applies the intrinsic default values defined in the previous chapter.

Changes of group policy values will be applied on the fly without needing to restart the software.

This software package contains registry files as samples for the convenience of the system administrators.

To apply a policy to users it can be published using this powershell command from the server:

```
Invoke-GPUpdate –Computer <computername> -Force -RandomDelayInMinutes
0
```

Or, from the client PC:

```
GPUpdate /force
```

An Update of the policy will restart the http servers. The Windows service will need to be restarted.

Appendix C – Unattended setup

The MSI packages of both programs can be installed without user interaction. This is useful for automatic software distribution systems like Microsoft System Center Configuration Manager.

Install-Command for the Desktop version:	msiexec /i "Busylight_HTTP_Desktop_Setup.msi" /q
Uninstall-Command for the Desktop version:	msiexec /x {D241853A-77CC-491D-9490-1A67EBE94F6A} /q
Install Command for the Windows Service version:	msiexec /i "kuando_HTTP_Service_Setup.msi" /q
Uninstall Command for the Windows Service version:	msiexec /x {45680EFF-9177-4DE4-B645-AC62C5D1BDBE} /q

The program GUID can change, please make sure you are using the right GUID. SCCM will do that automatically for you.

This is the content of a sample .reg file, defining http token and uri:

```
Windows Registry Editor Version 5.00


[HKEY_LOCAL_MACHINE\Software\Busylight]
"http_token"="tCcws9rrtEe6RkWAv4EPyw=="
"http_uri"=http://localhost:8989/
```

Alternatively, the settings can be done with PowerShell Commands:

```
new-item -Path HKLM:\Software\Busylight -ErrorAction SilentlyContinue

New-ItemProperty -Path HKLM:\SOFTWARE\Busylight -Name http_uri -PropertyType String -
Value "http://localhost:8989/" -Force

New-ItemProperty -Path HKLM:\SOFTWARE\Busylight -Name http_token -PropertyType String -
Value "tCcws9rrtEe6RkWAv4EPyw==" -Force
```